REMARKS

The examiner is thanked for the performance of a thorough search. By this amendment, Claims 1-3, 6, 10-12, 15, 19-23, 26, 30-34, 37, 41, and 42 are amended. Claims 43-52 are added. No claims are cancelled. Hence, Claims 1-52 are pending in the application.

The amendments to the claims as indicated herein do not add any new matter to this application. For example, new Claims 43-52 are similar to method Claims 10-20, except in the context of a system. Furthermore, amendments made to the original claims as indicated herein have been made to exclusively improve readability and clarity of the claims and not for the purpose of overcoming alleged prior art.

Each issue raised in the Office Action mailed November 9, 2006 is addressed hereinafter.

I. ISSUES RELATING TO PRIOR ART

Claims 1-42 are rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,008,814 to Mathur ("*Mathur*"), in view of U.S. Patent No. 6,535,924 to Kwok et al. ("*Kwok*"). The rejection is respectfully traversed.

A. CLAIM 1

Claim 1 recites:

> A method of software loading and initialization in a distributed network of nodes, the method comprising:
> storing, in a first storage of a master node, software packages and boot images, which software packages and boot images will be used by the nodes in the distributed network;
> **storing, in a second storage of the master node, preferred software version information and node type information** for each node in the distributed network;
> **receiving, at the master node, a request for a boot image and software packages from a node, in the distributed network, that is performing an initial boot;**
> **based on the request, retrieving preferred software version information of the node from the second storage;**
> **using the preferred software version information of the node, extracting a boot image and one or more software packages from the first storage;**
> delivering, to the node, the extracted boot image and one or more software packages;

.... (emphasis added)

At least the above-bolded features of Claim 1 are not taught or suggested by *Mathur* and *Kwok*, either individually or in combination.

1. High-level differences between Mathur *and present Claim 1*

Claim 1 recites a method for distributing a boot image and software packages to one or more nodes in a network. According to Claim 1, a request for a boot image and software packages is received from a node, in the network, that is performing an initial boot. Based on the request, preferred software version information of the node is retrieved from a particular storage associated with the master node. That software version information is used to extract a particular boot image and one or more software packages from a different storage associated with the master node. The master node delivers the extracted information to the node in order for the node to complete the initial boot.

In contrast, *Mathur* describes a process where an operator enters a "distribution command" at an input device of one of the nodes in a network (col. 5, lines 1-30). That distribution command includes a source field (which identifies a "source" node N_s that contains new software) and a destination field (which identifies destination nodes N_d that will receive the new software). Thus, one of the fundamental differences between Claim 1 and *Mathur* is that ***Mathur* discloses a process where a human operator of a "master" node initiates the update of another node, whereas Claim 1 recites a node, that is the recipient of a boot image and software packages, initiates the communication**. Thus, one of the benefits of the approach recited in Claim 1 is that a human operator is not required to initiate a boot process for other nodes in a network, nor is a human operator required to know the specifics of a node (e.g., how to specify the node, where the node is located, etc.) when a distribution of software occurs.

Because of this fundamental difference, *Mathur* and *Kwok* (individually and in combination) fail to teach or suggest numerous features of Claim 1.

2. Mathur *fails to teach or suggest the recited second storage*

On page 2, the Office Action equates the source node (N_s) or operator node (N_o) of *Mathur* with the "master node" of Claim 1. The Office Action then asserts that (1) col. 3, line 65 to col. 4, line 49; (2) col. 5, lines 3-30; and (3) col. 9, line 36 to col. 10, line 8 of *Mathur* teach "providing node information storage means on said master node for storing preferred software version information, node type...for each node in the network." This is incorrect. The corresponding language in present Claim 1 reads: "wherein the master node includes a second storage that stores preferred software version information and node type information for each node in the network."

Nothing in *Mathur* discloses that a source node (N_s) or operator node (N_O) includes a storage that stores **preferred software version information and node type information for each node in the network.** Col. 3, line 65 to col. 4, line 49 describes what contents a non-volatile storage device (NVSD) 103 for each node in a network may comprise. For example, a NVSD 103 of a node may be classified as "new," "dirty," "trial," or "old," depending on the software contained therein. However, no NVSD 103 stores preferred software version information or node type **for each node in the network.**

Col. 5, lines 3-30 states that a distribution command is entered at an N_o and that such a command contains a source field and a destination field. The source field identifies a N_s and the location of a NVSD 103 that contains new software. The destination field identifies nodes in the network that will receive the new software. Such nodes are referred to as destination nodes N_d. The destination field also contains location of NVSDs 103 within an N_d that will receive the new software. However, *Mathur* fails to disclose that the N_s or N_o includes a storage that stores

preferred software version information and node type **for each node in the network**. Additionally, the distribution command cannot be equated to the recited "second storage" because the distribution command does not store anything – it is simply a command. Furthermore, even if the distribution command could be considered a storage that stores information, the distribution command does not include **preferred software version information** *and* **node type information for each node in the network**. The distribution command only *identifies* nodes that will receive new system software. The distribution command does not even identify preferred software version information, much less node type information, of a node.

Col. 9, line 36 to col. 10, line 8 describes the general interactions between a master task of the N_s and slave tasks of a N_d. For example, the master task of the N_s requests information from slave tasks. In response, a slave task of a N_d send the requested information (such as which NVSD 103 of the N_d should receive the new software) to the master task. The master task selects destinations for a particular distribution and informs the slave tasks (of selected N_ds) of the number of packets that will be sent and "the identification (version and release number) of the system software to be distributed." Again, nothing in this cited portion of *Mathur* teaches or suggests that the N_s includes **a storage that stores preferred software version information and node type information for each node in the network**.

The Office Action, on page 3, further contends that "version and node identification in a list being stored for cutback after a failure in a *softload* reads on a node type and a trial version to be rolled back." However, the only "list" referred to in *Mathur* is a list of destinations in the destination field of a distribution command. There is no list in *Mathur* that is stored, much less a list that is stored for cutback after a failure in a softload process.

3. Mathur *fails to teach or suggest* "receiving, at the master node, a request for a boot image and software packages from a node"

The Office Action asserts that the message receiver 303 in FIG. 3 of *Mathur* discloses "a node performing an initial boot requests a boot program and software packages from said master node." This is incorrect. The corresponding language in present Claim 1 now reads "receiving, at said master node, a request for a boot image and software packages from a node."

Col. 10, lines 28-33 of *Mathur* states: "The message receiver 303 receives operator commands 304 and messages 305 from slave tasks in the network. In response to an operator command or a message, the message receiver 303 activates one of a plurality of processes in the message handler block 307." This portion of *Mathur*, as well as the surrounding text, fails to teach or suggest that a request **for a boot image and software packages** is received from a node that is performing an initial boot. Indeed, as indicated above, according to present Claim 1, a node in a network that is to receive a boot image and software packages initiates the communication with the recited request. In contrast, *Mathur* teaches that "the communication is **initiated by an operator command...to the master task of ...the source node N_s** of a softload process" (col. 11, lines 16-21). Col. 4, line 64 to col. 5, line 3 of *Mathur* further states:

> After the new version of system software is installed in the source node N_s, the distribution of the new system software to the other nodes is initiated. Typically, basic operation of the new system software is tested before it is distributed. Advantageously, **the distribution is initiated by a "distribution command" which is entered by the operator at an input/output device in one of the nodes.** (emphasis added)

Therefore, not only does *Mathur* fail to disclose "receiving a request for a boot image and software packages from a node," *Mathur* **teaches away** from this feature of present Claim 1.

4. Mathur *fails to teach or suggest "based on the request, retrieving preferred software version information of the node from the second storage"*

The Office Action cites (1) col. 5, lines 3-30; (2) col. 9, line 36 to col. 10, line 8; and (3) col. 7, lines 24-44 of *Mathur* for disclosing "retrieving said node's preferred software version information from said node information storage means." This is incorrect. The corresponding language in present Claim 1 now reads "based on the request, retrieving preferred software version information of the node from the second storage."

As discussed above, *Mathur* fails to teach or suggest (1) the recited "second storage" and (2) the recited request for a boot image and software packages from a node that is the recipient of a boot image and software packages. Therefore, *Mathur* also fails to teach or suggest "**based on the request**, retrieving preferred software version information of the node from the **second storage**" (emphasis added).

Col. 5, lines 3-30 and col. 9, line 36 to col. 10, line 8 of *Mathur* have been discussed in detail above with respect to the recited "second storage." Both cited portions of *Mathur* also lack any teaching or suggestion of the recited request for a boot image and software packages from a node that is performing an initial boot.

Col. 7, lines 24-44 of *Mathur* describes what checks a destination node (N_d) performs in response to receiving a cutover command after a successful distribution. Specifically, an N_d checks (1) whether the cutover process should proceed and then (2) whether the NVSD 103 of the N_d has a status of "new"; in which case a consistency check is performed on the NVSD 103. If the consistency check is successful, then the status of the NVSD 103 is changed to "trial", after which the N_d performs an initial program load (IPL). This cited portion of *Mathur* also fails to teach or suggest anything relating to retrieving **preferred software version information of the node**, much less retrieving, based on the recited request, such information **from a storage that stores preferred software version information for each node in a network**.

5. Mathur *fails to teach or suggest "using the preferred software version information of the node, extracting a boot image and one or more software packages from the first storage"*

The Office Action cites (1) col. 5, lines 31 to col. 6, line 54 and (2) step 202 of FIG. 2 of *Mathur* for disclosing "extracting boot program and software packages from said software package storage means using said node's preferred software version information." This is incorrect. The corresponding language in present Claim 1 now reads "using the preferred software version information of the node, extracting a boot image and one or more software packages from the first storage."

Step 202 of FIG. 2 and col. 5, line 31 to col. 6, line 2 of *Mathur* disclose that a consistency check is performed with respect to the N_s and a N_d. The consistency check with respect to the N_s is to verify the new software. The consistency check with respect to a N_d checks (1) whether there is a NVSD 103 at the specified location of the N_d and (2) whether data can be written to the specified NVSD 103. These cited portions of *Mathur* fail to teach that a boot image and one or more software packages are extracted from a storage **using the preferred software version information of a node**, much less the preferred software version information retrieved in the previous step. Indeed, nothing in these cited portions of *Mathur* can be equated to extracting preferred software version information from a storage on the N_s because they relate only to preliminary matters before distribution and not yet to extracting software.

The remaining cited portion of *Mathur* (col. 6, lines 3-54) describes the process of a software distribution from the N_s to a N_d. For example, the new software is transmitted in one or more batches of data packets that each contains a checksum value of its data. Also, the N_s retains information concerning the progress of distribution. This remaining cited portion of Mathur fails to teach or suggest that a boot image and one or more software packages are

extracted from the recited first storage **using the preferred software version information, of a node**, that was retrieved from the recited second storage.

One reason for this lack of teaching is that the invention of *Mathur* depends on (1) a human operator specifying destination nodes (N_d) for a distribution and (2) each destination node containing logic for handling distributions based on a status for each NVSD 103 with which the destination node is associated. In contrast, present Claim 1 recites that a master node stores an association (1) between a node and preferred software version information (i.e., second storage) and (2) between preferred software version information and a particular boot image and set of software packages (i.e., first storage). There are no similar associations in *Mathur*.

Based on the foregoing, *Mathur* fails to teach or suggest numerous features of present Claim 1. The Office Action does not allege that *Kwok* discloses those features. Therefore, it is respectfully submitted that *Mathur* and *Kwok* fail to teach or suggest (individually or in combination) all the features of present Claim 1. Removal of the 35 U.S.C. § 103(a) rejection with respect to present Claim 1 is respectfully requested.

B. CLAIMS 10, 21, 32, AND 43

Each of independent Claims 10, 21, 32, and 43 is either a method, a computer-readable storage medium, an apparatus, or a system claim. Each of Claims 10, 21, 32, and 43 recite features discussed above that distinguish present Claim 1 over *Mathur* and *Kwok*. Therefore, each of Claims 10, 21, 32, and 43 is allowable for the reasons given above with respect to present Claim 1.

C. CLAIMS 7, 16, 27, 38 AND 48

Each of Claims 7, 16, 27, 38 and 48 is dependent upon one of the independent claims discussed above, and additionally recites "wherein the **master node** has the ability to **categorize**

nodes into classes where all of the nodes in a particular class of nodes have the same software configuration and may have differing processor types" (emphasis added). The Office Action cites col. 5, lines 3-27 of *Mathur* for disclosing this feature. This is incorrect. The Office Action focuses on the following paragraph of that cited portion, which states:

> Advantageously, the system is implemented so that a distribution command is recognizable by a controller 100 only when it is issued with predefined access privileges. Furthermore, **such privileges may be defined with an hierarchical structure** so that the privilege required for a distribution process depends upon the importance of the software. (emphasis added)

The Office Action asserts that the "structure representing grouping of nodes according to some particular privilege configuration reads on classes of node of same configuration." It is respectfully submitted that there is no recitation of a "grouping of nodes" in *Mathur* nor does the "hierarchical structure" of *Mathur* represent a grouping or class of nodes. Instead, the hierarchical structure **defines access privileges**. Additionally, the only references to a "configuration" in *Mathur* is to the configuration of a network in terms of which nodes are connected (col. 3, lines 48-51) and how they are connected (col. 1, lines 33-35). There is no teaching of nodes in the network that "have the same software configuration", as recited in present Claim 1.

Furthermore, *Mathur* fails to teach or suggest that a **N_s or N_o** (i.e., the alleged master node) **has the ability to categorize nodes into classes**. The only "abilities" of a N_s or N_o are (1) performing a consistency check of an associated NVSD 103, (2) distributing new software to destination nodes N_d that are identified in a distribution command, and (3) relaying progress information, of a distribution, to an operator. Therefore, the N_s and N_o do not have the ability to categorize nodes into classes where all nodes in a particular class have the same software configuration.

Based on the foregoing, *Mathur* and *Kwok* fail to teach or suggest (individually or in combination) the features of Claims 7, 16, 27, 38 and 48. Removal of the 35 U.S.C. § 103(a) rejection with respect to each of Claims 7, 16, 27, and 38 is therefore respectfully requested.

D. REMAINING DEPENDENT CLAIMS

The dependent claims not discussed thus far are dependent claims, each of which depends (directly or indirectly) on one of the independent claims discussed above. Each of the dependent claims is therefore allowable for the reasons given above for the claim on which it depends. In addition, each of the dependent claims introduces one or more additional limitations that independently render it patentable. However, due to the fundamental differences already identified, to expedite the positive resolution of this case, a separate discussion of those limitations is not included at this time. The Applicant reserves the right to further point out the differences between the cited art and the novel features recited in the dependent claims.

II. CONCLUSIONS & MISCELLANEOUS

For the reasons set forth above, all of the pending claims are now in condition for allowance. The Examiner is respectfully requested to contact the undersigned by telephone relating to any issue that would advance examination of the present application.

A petition for extension of time, to the extent necessary to make this reply timely filed, is hereby made. If applicable, a law firm check for the petition for extension of time fee is enclosed herewith. If any applicable fee is missing or insufficient, throughout the pendency of this application, the Commissioner is hereby authorized to any applicable fees and to credit any overpayments to our Deposit Account No. 50-1302.

Respectfully submitted,
HICKMAN PALERMO TRUONG & BECKER LLP

Dated: February 9, 2007

Daniel D. Ledesma
Reg. No. 57,181

2055 Gateway Place Suite 550
San Jose, California 95110-1093
Telephone No.: (408) 414-1080
Facsimile No.: (408) 414-1076